Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007)

Feature Selection for Change Detection in Multivariate Time-Series

Michael Botsch
Institute for Circuit Theory and Signal Processing
Technical University Munich
80333 Munich, Germany
Email: botsch@tum.de

Josef A. Nossek
Institute for Circuit Theory and Signal Processing
Technical University Munich
80333 Munich, Germany
Email: josef.a.nossek@tum.de

Abstract— **In machine learning the preprocessing of the observations and the resulting features are one of the most important factors for the performance of the final system. In this paper a method to perform feature selection for change detection in multivariate time-series is presented. Feature selection aims to determine a small subset which is representative for the change detection task from a given set of features. We are dealing with time-series where the classification has to be done on time-stamp level, although the smallest independent entity is a scenario consisting of one or more time-series. Despite this difficulty we will show how feature selection based on the generalization ability of a classifier can be realized by defining a cost function on scenario level. For the classification step in the feature selection process a modified *Random Forest* (RF) algorithm—which we will call *Scenario Based Random Forest* (SBRF)—is used due to its intrinsic possibility to estimate the generalization error. The excellent performance of the proposed feature selection algorithm will be shown in a car crash detection application.**

I. Introduction

Time-series have been studied extensively mainly in four important areas of application [1]: forecasting, estimating the transfer function of a system, analysis of effects of unusual intervention events to a system and discrete control systems. Among the most successful techniques applied thereby is modeling the time-series as an autoregressive integrated moving average stochastic process. In this paper we will not determine the parameters of stochastic models for time-series, but rather machine learning techniques will be applied in order to classify new time-series based on the observation of past time-series and their labeling. In particular we focus on the feature selection task since the preprocessing and the resulting features are one of the most important factors for the classification performance.

Multivariate time-series classification and change detection in time-series do not easily fit in the usual feature-value model used in machine learning, since the scenarios to be classified consist of a large sequence of elementary features. Whereas for time-series this elementary features are the successive values of the signals to be classified, the same problem appears also in sequence recognition, e. g., pattern recognition in DNA sequences or character sequences in a written text. In order to apply feature-value machine learning algorithms to this kind of problems one has to change the representation of the scenarios to be classified from the elementary features to high-level features which can represent the desired classes better and which can be computed from the elementary features. This is due to the fact, that an intrinsic property of time-series or other sequences is the statistical dependency of adjacent values. Since the nature of this dependence is of practical interest, the aim in feature extraction is to find those high-level features which describe best the characteristic features for each class. The construction of these high-level features normally requires *a priori* knowledge about the process that generated the data. If such domain knowledge is not available, a large set of predefined high-level features can be constructed from the data and in a following step feature selection can be performed in order to keep only a small set of high-level features which are best suited for the classification task. In the paper we will mainly focus on the latter task.

A reduced number of features not only combats the curse of dimensionality [2] but also leads to a reduced computational complexity. Feature selection techniques can be divided into filter, wrapper, and embedded methods. We will focus on wrapper and embedded methods where a learning machine is involved in the selection step and the importance of the features is estimated based on the performance of the learning machine. An advantage of wrapper and embedded methods is that they also take into account the interactions among the features when evaluating their importance for the classification task. As learning machine we will use a modified version of the *Random Forest* (RF) algorithm [3] which will be extended, such that the misclassification error rate can be estimated on scenario-level. We will call the algorithm *Scenario-Based Random Forest* (SBRF). In the paper we will perform forward-backward feature elimination. The forward feature selection is accomplished by the use of decision trees in the SBRF and the backward feature elimination is realized by starting with a set of features and reducing this set based on the performance of the SBRF.

Time-series classification using machine learning has already been discussed in some recent works, e. g., in [4], [5], [6], [7], [8]. Also possibilities to perform feature extraction from time-series have been proposed in [9], [10], [11]. However, none of this works considers the change detection case. In change detection some segments of a time-series have one class label and other segments have different class labels.

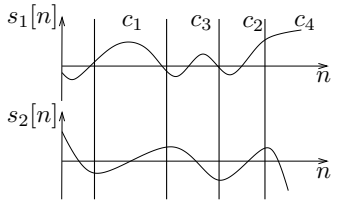

Fig. 1: Change Detection

Moreover, in change detection at a certain time instance not the whole time-series is available but only the sequence up to this point. The main contribution of this paper is to present an efficient machine learning algorithm for performing feature selection for change detection in multivariate time-series. The method is also applicable to the static time-series classification where the class label is not changing with time.

Throughout the paper, vectors and matrices are denoted by lower and upper case bold letters, and random variables and random processes are written using *sans serif* fonts.

II. The Change Detection Problem

In classification and regression tasks machine learning procedures aim to estimate the values of a target attribute v given a set of measured observation attributes $\boldsymbol{u}$. In the change detection problem the observations are realizations of random processes—in the following called scenarios—denoted with $\boldsymbol{S}$. In the multivariate case a scenario $\boldsymbol{S} \in \mathbb{R}^{L \times n_{\text{end}}}$ consists of L time-series and has the length n_{end} whereas the target attributes taking on discrete values c_k, $k = 1, \ldots, K$—called classes or labels—are also realizations of the random process $\boldsymbol{y} \in \{c_1, \ldots, c_K\}^{1 \times n_{\text{end}}}$:

$$\boldsymbol{S} = [\boldsymbol{s}[1], \ldots, \boldsymbol{s}[n_{\text{end}}]], \text{ with} \quad (1)$$
$$\boldsymbol{s}[n] = [\mathsf{s}_1[n], \ldots, \mathsf{s}_L[n]]^{\mathrm{T}} \in \mathbb{R}^L \quad \text{and}$$
$$\boldsymbol{y} = [\mathsf{y}[1], \ldots, \mathsf{y}[n_{\text{end}}]]. \quad (2)$$

Unlike the usual feature-value model where the observations $\boldsymbol{u}_m$ are assumed to be realizations of i.i.d. random variables, here the random variables from the same scenario, e. g., $\boldsymbol{s}[n]$ and $\boldsymbol{s}[n-1]$ are dependent. Thus, the smallest entity that can be assumed independent from the others is an entire scenario $\boldsymbol{S}$. The set $\mathcal{T}$ of measured observations—called training set—consists of M scenarios and the corresponding target values

$$\mathcal{T} = \{(\boldsymbol{S}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{S}_M, \boldsymbol{y}_M)\}. \quad (3)$$

One of the difficulties that appears in change detection is that a scenario $\boldsymbol{S}_m$ consists of segments where the corresponding target labels change. In most of the literature on time-series classification using machine learning it is assumed that in a scenario $\boldsymbol{S}_m$ the class label $\mathsf{y}[n]$ is constant, i. e., a scenario belongs to one class. This is not the case in the change detection task. As an example in Fig. 1 a scenario consisting of $L = 2$ time-series is shown and the scenario is divided in 4 segments with the labels c_1, c_2, c_3 and c_4.

The aim of change detection is to figure out for a new scenario which label the current segment has at a given time-index n by using only the information from the time-series up to this point. With $\boldsymbol{S}^{[1,n]} = [\boldsymbol{s}[1], \ldots, \boldsymbol{s}[n]] \in \mathbb{R}^{L \times n}$ denoting the time-series up to time-index n we are looking for the mapping

$$g_n : \mathbb{R}^{L \times n} \to \{c_1, \ldots, c_K\},\ \boldsymbol{S}^{[1,n]} \mapsto \mathsf{y}[n]. \quad (4)$$

In order to apply machine learning algorithms one can extract from $\boldsymbol{S}^{[1,n]}$ high-level features, e. g., slope, duration of local maxima, magnitude of a peak relative to its neighborhood, peak counts, fraction of energy between two frequencies, etc.. These high-level features are then stored in the vector $\boldsymbol{x}[n] \in \mathbb{R}^N$. Based on the input vector $\boldsymbol{x}[n]$ and the target values $\mathsf{y}[n]$ a machine learning algorithm can then be used to learn the mapping

$$f : \mathbb{R}^N \to \{c_1, \ldots, c_K\},\ \boldsymbol{x}[n] \mapsto \mathsf{y}[n]. \quad (5)$$

Thus, instead of searching for the mappings g_n, the task changes into finding the mapping f using a machine learning algorithm. However, this is not a classical classification problem since the successive vectors $\boldsymbol{x}[n]$ from the same scenario are strongly correlated. Moreover, the cost function used for the classification must be changed because the evaluation of the change detection performance will be done on scenario level and not on time-stamp level. Details will be presented in Section IV. As a consequence also the validation must be performed on scenario level. For example in V-fold cross validation all vectors $\boldsymbol{x}[n]$ from a scenario must be either in the training or in the validation set.

The step of changing the representation of the scenario at time instance n from $\boldsymbol{S}^{[1,n]}$ to $\boldsymbol{x}[n]$ is crucial for the final performance of the change detection system. This step can be decomposed into a feature generation and feature selection task which will be discussed in the following.

III. Feature Generation

The aim of feature generation is to construct high-level features from time-series. Normally this step requires *a priori* knowledge about the process that generated the data. If one has such *a priori* information, features should be constructed that capture the known properties of the data. However, it might be that a previous filtering of the data improves the discrimination property of the generated high-level features. Thus, one should also take into consideration the high-level features resulting after preprocessing steps and then figure out in the feature selection task which features are best suited for the classification.

In some applications where a problem specific modeling is not possible or not desired, one approach is to construct a large set of possible templates for high-level features [7], [12], [13]. In [13] for example structure detectors are introduced which try to fit a function (e. g., constant, exponential, triangular) with free parameters to a time-series such that the difference between the raw data and the function is minimized. Other high-level features that have been suggested in the technical literature are summarized in [14].

One of the most powerful methods for time-series classification use similarity measures which are defined to consider

the time specific nature of the data and then classifiers using this distance measure (e. g., nearest neighbor algorithm) are used [15]. Assuming that a set of templates for the scenarios or for sections of the scenarios is known, methods like [16] *Dynamic Time Warping* (DTW) can be applied and the resulting similarities to the templates can be used as high-level features of the considered scenario.

In the feature generation step the mapping

$$h_n : \mathbb{R}^{L\times n} \to \mathbb{R}^{\tilde{N}},\ \boldsymbol{S}^{[1,n]} \mapsto \tilde{\boldsymbol{x}}[n] \tag{6}$$

is performed, where the vector $\tilde{\boldsymbol{x}}[n] \in \mathbb{R}^{\tilde{N}}$ consists of high-level features. It is important to note here, that the dimensionality of the vector $\tilde{\boldsymbol{x}}[n]$ is not changing with time. If for example the ith entry in $\tilde{\boldsymbol{x}}[n]$ stands for the amplitude of the first local minima in the ℓth time-series, its value will be zero until the first local minima is included in $\boldsymbol{S}^{[1,n]}$. Since the dimensionality $\tilde{N}$ of $\tilde{\boldsymbol{x}}[n]$ will be large and many entries may contain redundant information, the next step—feature selection—aims to reduce $\tilde{\boldsymbol{x}}[n]$ to the vector $\boldsymbol{x}[n]$ with smallest possible dimensionality N that is required for an accurate classification.

IV. Feature Selection

In the following a forward-backward feature selection will be considered. We assume that a large set of high-level features has been extracted from the time-series and now the task is to figure out which of these features are best suited for the classification task. Thus, we are looking for the selection matrix $\boldsymbol{J} \in \{0,1\}^{N\times\tilde{N}}$, $N < \tilde{N}$ such that

$$\boldsymbol{x}[n] = \boldsymbol{J}\tilde{\boldsymbol{x}}[n]. \tag{7}$$

Feature selection is strongly related to the task of model assessment and thus also the methods are similar, i. e., either statistical tests or machine learning methods like V-fold cross validation can be used. In this work we will focus on machine learning methods and perform the selection based on the estimated generalization error of a powerful classification algorithm.

Since we are dealing with time series, first we have to define what the expected risk—or classification error—is in change detection. If the corresponding targets to the time-series $\boldsymbol{S}$ are accumulated in the vector $\boldsymbol{y}$, the expected risk is

$$R(\boldsymbol{g}) = \mathrm{E}_{\boldsymbol{S},\boldsymbol{y}}\left\{L(\boldsymbol{g}(\boldsymbol{S}),\boldsymbol{y})\right\}, \tag{8}$$

and the empirical risk is

$$R_{\text{emp}}(\boldsymbol{g}) = \frac{1}{M}\sum_{m=1}^{M} L(\boldsymbol{g}(\boldsymbol{S}_m),\boldsymbol{y}_m), \tag{9}$$

where $L(\boldsymbol{g}(\boldsymbol{S}),\boldsymbol{y})$ denotes the loss function with $\boldsymbol{g}(\boldsymbol{S}) \in \{c_1,\ldots,c_K\}^{1\times n_{\text{end}}}$,

$$\boldsymbol{g}(\boldsymbol{S}) = \left[g_1\left(\boldsymbol{S}^{[1,1]}\right),\ldots,g_{n_{\text{end}}}\left(\boldsymbol{S}^{[1,n_{\text{end}}]}\right)\right], \tag{10}$$

and $\mathrm{E}_{\boldsymbol{S},\boldsymbol{y}}\left\{L(\boldsymbol{g}(\boldsymbol{S}),\boldsymbol{y})\right\}$ is the expectation of $L(\boldsymbol{g}(\boldsymbol{S}),\boldsymbol{y})$ with respect to the random processes $\boldsymbol{S}$ and $\boldsymbol{y}$. Since with the mappings h_n we transform the scenario $\boldsymbol{S}$ to its high-level representation

$$\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}[1],\ldots,\tilde{\boldsymbol{x}}[n_{\text{end}}]] \in \mathbb{R}^{\tilde{N}\times n_{\text{end}}} \tag{11}$$

and with the selection matrix $\boldsymbol{J}$ from Eq. (7) we obtain

$$\boldsymbol{X} = \boldsymbol{J}\tilde{\boldsymbol{X}} = [\boldsymbol{x}[1],\ldots,\boldsymbol{x}[n_{\text{end}}]] \in \mathbb{R}^{N\times n_{\text{end}}}, \tag{12}$$

the expected risk $R(f(\boldsymbol{J}))$ and the empirical risk $R_{\text{emp}}(f(\boldsymbol{J}))$ can now be introduced analogously to Eq. (8) and Eq. (9) by replacing $\boldsymbol{g}(\boldsymbol{S})$ with

$$\boldsymbol{f}(\boldsymbol{X}) = \boldsymbol{f}(\boldsymbol{J}\tilde{\boldsymbol{X}}) = \left[f\left(\boldsymbol{x}[1]\right),\ldots,f\left(\boldsymbol{x}[n_{\text{end}}]\right)\right]. \tag{13}$$

For the evaluation of the classifier performance it does not matter whether we work with elementary or high-level features such that the loss function must be chosen to fulfil

$$R(f(\boldsymbol{J})) = R(\boldsymbol{g}) \quad \text{and} \quad R_{\text{emp}}(f(\boldsymbol{J})) = R_{\text{emp}}(\boldsymbol{g}). \tag{14}$$

In the static time-series classification the loss function only measures whether the time-series has been assigned to the correct class and in the change detection task the time instance when a class-change is identified or missed determines the loss function, such that it does not play any role whether elementary or high-level features are used.

In order to obtain a good change detection system, one has to find the mappings g_n that minimize the expected loss $R(\boldsymbol{g})$, or according to Eq. (14) the mapping $f(\boldsymbol{J})$ which minimizes $R(f(\boldsymbol{J}))$. For the feature selection task we assume that the only mapping we do not know and which influences $R(f(\boldsymbol{J}))$ is the selection matrix $\boldsymbol{J}$. Our aim is to obtain a small expected risk and thus we implement the mapping f by using a powerful classification algorithm such that the relevant quantity in changing $R(f(\boldsymbol{J}))$ is the matrix $\boldsymbol{J}$. Thus, the feature selection task can be formulated as finding the selection matrix $\boldsymbol{J}$

$$\boldsymbol{J} = \underset{\tilde{\boldsymbol{J}}}{\operatorname{argmin}}\left\{R(f(\tilde{\boldsymbol{J}}))\right\}. \tag{15}$$

Since the probability density functions needed to perform this optimization task are not known, one uses $R_{\text{emp}}(f(\boldsymbol{J}))$ for training the classifier and estimates $R(f(\boldsymbol{J}))$ from the M observed scenarios with methods like V-fold cross validation or the *out-of-box* (oob) technique [17], [18], [19]. We will use the latter in this work since oob-estimation can easily be integrated in the SBRF-algorithm which we will apply to implement the mapping f. In [17] Breiman gives empirical evidence that the oob-estimate is as accurate as using a test set of the same size as the training set.

The only missing part now in order to realize feature selection for change detection is the definition of the loss function $L(\boldsymbol{f}(\boldsymbol{X}),\boldsymbol{y})$. The loss function should be designed such that it takes account of the importance that a misclassification has at the various time instances of a scenario. If this importance is stored in the vector

$$\boldsymbol{\gamma} = [\gamma[1],\ldots,\gamma[n_{\text{end}}]] \in \mathbb{R}^{1\times n_{\text{end}}} \tag{16}$$

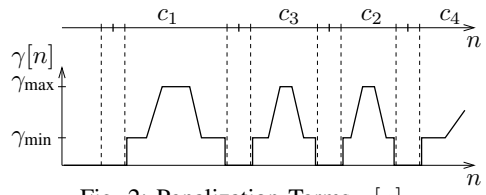

Fig. 2: Penalization Terms $\gamma[n]$

and the time instances when a class change occurs either in $\boldsymbol{y}$ or in $\boldsymbol{f}(\boldsymbol{X})$ are denoted with $n_{\mathrm{cd},i}$, then we define

$$L(\boldsymbol{f}(\boldsymbol{X}),\boldsymbol{y})=\frac{1}{C}\sum_{i=1}^{C}\gamma[n_{\mathrm{cd},i}](1-I(f(\boldsymbol{x}[n_{\mathrm{cd},i}]),\mathsf{y}[n_{\mathrm{cd},i}])), \quad (17)$$

where C is the total number of changes in $\boldsymbol{y}$ and $\boldsymbol{f}(\boldsymbol{X})$, and the function $I(a,b)$, with $a\in\mathbb{R}$ and $b\in\mathbb{R}$ is defined as

$$I(a,b)=\begin{cases}1 & \text{if } a=b\\ 0 & \text{else}\end{cases}. \quad (18)$$

Note that by setting $\gamma[n]=0$ for all time-stamps where an observed scenario has a length smaller than n_{end} we can consider scenarios of different lengths. Another possible definition of the loss function is

$$L(\boldsymbol{f}(\boldsymbol{X}),\boldsymbol{y})=\frac{\sum_{n=1}^{n_{\mathrm{end}}}\gamma[n](1-I(f(\boldsymbol{x}[n]),\mathsf{y}[n]))}{\sum_{n=1}^{n_{\mathrm{end}}}\gamma[n]}, \quad (19)$$

but this definition does not differentiate whether misclassifications are caused by the same or a different event in the scenario, thereby putting too much emphasis on misclassifications that have the same reason. Thus, in the following the loss from Eq. (17) will be used.

The importance $\boldsymbol{\gamma}$ for the example from Fig. 1 may have the values shown in Fig. 2. The values of $\gamma[n]$ have been chosen here to penalize a misclassification strongly if it occurs in regions that clearly belong to one of the classes while the penalization gets smaller the closer one moves towards a change to a different class. Because in time-series adjacent values are strongly related usually a certain time-interval is required in order to construct a high-level feature, e. g., a local maximum can only be recognized as a maximum after observing some values after the peak. This is the reason why in the vicinity of a class-change a "do not care" interval is introduced i. e., $\gamma[n]=0$ for all n in this interval such that a misclassification here is not taken into consideration when computing the risk. It should be noted that this requires to set the corresponding value $n_{\mathrm{cd},i}$ from Eq. (17) to the next time instance following the "do not care" interval.

In the next subsection we will review how the RF algorithm can be used for feature selection in the usual feature-value model and in Subsection IV-B we show how this algorithm must be modified in order to obtain the SBRF-algorithm which can be used for feature selection in change detection tasks.

A. Feature Selection with RF

The RF algorithm has been introduced by Breiman in [3] and it is one of the most powerful known classification algorithms. It is a randomized and aggregated version of the well-known [20] *Classification And Regression Tree* (CART) algorithm strengthened by the bagging (stands for *b*ootstrap *agg*regating) technique. Given a set of input vectors $\boldsymbol{u}$ and the corresponding targets v, the idea underlying the RF algorithm is to construct a large number B of classifiers $D_i(\boldsymbol{u})$, $i=1,\ldots,B$, with low bias, e. g., full grown decision trees and then to take a majority vote among the individual classifiers. It is proven in [3] that the algorithm does not overfit as more trees are added to the RF. The step of taking the majority vote for classification tasks or the average in regression problems reduces the variance of the classifier determined using the aggregated algorithm considerably without increasing its bias compared to the bias and variance of the individual classifiers in the ensemble [21]. Thus, one obtains a classifier with low bias and low variance leading to a small generalization error. It is interesting to note that for classification the generalization error does not decompose into a sum of bias and variance as it is the case in regression, but in a fraction term containing the bias and variance, such that it is possible to reach the minimum of the generalization error by only decreasing the variance [22]. This explains why ensemble methods like AdaBoost or RF perform so well in classification.

In order to decrease the variance term it is important that the individual classifiers differ from each other, i. e., have a low correlation. To achieve this goal a randomization source is introduced in the construction of each tree. Although there are many possibilities to introduce randomness, for the feature selection task the bagging technique is crucial allowing the oob-estimation of the generalization error [3]. The idea underlying the oob-estimate is that each tree-classifier in the ensemble is built based on a training set that is obtained by applying the bootstrap procedure, i. e., sampling with replacement from the original training set. Assuming that there are M' feature-value patterns in the original training set, there are $\approx(1-1/M')^{M'}\cdot M'$ ($\approx e^{-1}M'$ for large M') patterns that are not used in the learning phase of the ith tree-classifier. On the other hand, this means that every pattern from the training set has not been used for the training of $\approx 36\%$ of the B trees in the RF. Thus, one can estimate the generalization error by taking for each pattern the majority vote only among those trees which have not seen this pattern during the training phase.

Being able to obtain an honest estimate of the generalization error using the oob-technique the importance of a variable can be determined by comparing the performance of the RF on the original data with the performance when the information in the variable is removed. Firstly, one computes the oob-estimate of the generalization error e'_{oob}. Then, assuming that the importance of the jth variable is analyzed, for each of the B trees in the RF the information of the jth variable is removed by randomly permuting the jth entry of the input vectors $\boldsymbol{u}$ which have not been used in the training phase by the tree. Now the generalization error $e'^{(j)}_{\mathrm{oob}}$ is estimated again using the oob-method. The difference

$$\Delta'^{(j)}=e'^{(j)}_{\mathrm{oob}}-e'_{\mathrm{oob}} \quad (20)$$

represents an importance measure for the jth variable. The larger $\Delta'^{(j)}$, the more important the jth variable is for the classification. This importance measure can be applied to every learning machine, for example by estimating the generalization error using V-fold cross-validation. Thus, this procedure is a wrapper method for feature selection.

In [23] Breiman also suggests another importance measure which is typical for RF and thus leads to an embedded method. It makes use of the margin that can be calculated using the RF algorithm. Assuming that the example $(\boldsymbol{u}, \mathsf{v})$ has not been used for the training of B' trees, the margin is defined as [3]

$$\tilde{mg}(\boldsymbol{u}, \mathsf{v}) = \frac{1}{B'} \sum_{i=1}^{B'} I(D_i(\boldsymbol{u}), \mathsf{v}) - \max_{c_k \neq \mathsf{v}} \left\{ \frac{1}{B'} \sum_{i=1}^{B'} I(D_i(\boldsymbol{u}), c_k) \right\} \tag{21}$$

where c_k denotes the kth class. A positive margin means that the example has been classified correctly, whereas a negative margin occurs for misclassifications. The margin offers a measure of confidence in the classification result. In order to figure out the importance of the jth variable the margin can be used. Thereby, the increase in the average margin over all data examples when removing the information from the jth entry in $\boldsymbol{u}$ is taken as a measure of importance of feature j.

After computing the importance for each feature one can remove the features which have little or no importance, thereby reducing the dimensionality of the input vector $\boldsymbol{u}$. Then, a new RF is built using only the important features. If the generalization error estimated using the new RF is not worse than the one estimated using all features, then a smaller set of features has been found for the classification task. Thus, in an iterative process one can figure out a small subset of features that still leads to the desired classification performance.

It is important to note that the RF algorithm leads to accurate classification results even if the dimensionality of the input vector $\boldsymbol{u}$ is high [3]. This is the reason why in the 2003 NIPS competition on feature selection in high-dimensional data the top entries performed feature selection using the RF algorithm.

B. Feature Selection for Time-Series

When dealing with time-series one must take account for the fact that the loss function is defined on scenario level (see Eq. (17)). Thus, to apply the feature selection described in the previous subsection to time-series some changes are necessary in the RF algorithm leading to the SBRF algorithm. Firstly, in the SBRF algorithm the bootstrap method used when constructing a forest must be done on scenario level, i. e., the training set $\mathcal{T}_i'$ needed for the ith tree is constructed by sampling M scenarios in their high-level feature representation $\tilde{\boldsymbol{X}}$ with replacement from $\mathcal{T}$. This allows the computation of an honest oob-estimate $R_{\text{oob}}(f(\boldsymbol{J}))$ of the risk $R(f(\boldsymbol{J}))$ since each scenario has not been used in the training phase by $\approx 0.36\% B$ trees in the forest.

Secondly, one must consider the varying penalization weight of a misclassification represented by the importances $\boldsymbol{\gamma}$ in every scenario. This can be realized by oversampling the patterns with a high corresponding value $\gamma[n]$. The idea here is based on the methods applied for imbalanced data-sets where one class constitutes only a very small minority of the data [24]. In these cases one can either assign a high cost to the misclassification of the minority class and then minimize the overall cost [25] or sampling techniques can be applied. Different sampling methods exist, e. g., downsampling the majority class, oversampling of the minority class [26] or both. Oversampling raises the weight of those samples that are replicated and this is exactly what we want. More precisely we will replicate the examples $(\tilde{\boldsymbol{x}}[n], \mathsf{y}[n])$ according to their weighting $\gamma[n]$. This leads to the training set $\mathcal{T}_i$ which is used to construct the ith tree in the SBRF. At this level there is no difference to the RF algorithm, i. e., unpruned trees are grown where at each node only a reduced number of features is examined for the best split thereby introducing an additional source of randomization.

The loss function for the construction of a tree D is

$$L_D(D(\boldsymbol{x}'), \mathsf{y}') = 1 - I(D(\boldsymbol{x}'), \mathsf{y}'), \tag{22}$$

where the realizations of $\boldsymbol{x}'$ and y' used to built the ith tree are stored in $\mathcal{T}_i$. The minimization of the expectation of $L_D(D(\boldsymbol{X}), \boldsymbol{y})$ also assures the minimization of the risk using the loss from Eq. (17) or Eq. (19).

After building B trees we can compute for each time instance of a new scenario the class label by taking the majority vote among the trees, thereby implementing the function f.

To find out the importance of the jth high-level feature one proceeds as in the usual feature-value model from the previous subsection, namely by computing the loss on performance due to the jth feature

$$\Delta^{(j)} = R_{\text{oob}}^{(j)}(f(\boldsymbol{J})) - R_{\text{oob}}(f(\boldsymbol{J})). \tag{23}$$

Thereby, $R_{\text{oob}}^{(j)}(f(\boldsymbol{J}))$ is the oob-estimate of the risk when the information from the jth feature has been removed. Alg. 1 sums up the steps required to perform feature selection using the SBRF algorithm.

The larger the value of $\Delta^{(j)}$ the more important the jth feature is. Since only the relative magnitudes of $\Delta^{(j)}$ are of interest one can normalize these quantities such that the most important feature has the importance 100.

Then, in an iterative process one can eliminate the features with low importance as long as the estimated risk is still acceptable, thereby making a hypothesis about the features to use. Since for every new hypothesis a new SBRF is required one can speed up the selection process by eliminating—especially at the first iterations—more than one feature in line 17 of Alg. 1.

A consequence of the random choice of the $\sqrt{\tilde{N}}$ features that are examined for the best split in a tree (see line 10 of Alg. 1) is the fact that masking effects do not appear as in the CART algorithm [20]. Thus, features that share much of the same important information will lead to high values of the corresponding $\Delta^{(j)}$ making it difficult to eliminate such redundant features. Fortunately, this effect only plays a

role in the final selection iterations. Therefore, when only a small number of features is left, one can extend the algorithm and test whether the removal of a different feature than the one with lowest estimated importance can still fulfill the requirement of having a generalization error below the critical value from line 3 of Alg. 1. In line 12 of Alg. 1 the estimate

Algorithm 1 SBRF feature selection

$\boldsymbol{J} = \boldsymbol{I}_{\tilde{N}}$
2: $R_{\text{oob}}(f(\boldsymbol{J})) = 0$
while $R_{\text{oob}}(f(\boldsymbol{J})) \leq$ critical value **do**
4: **for** $m = 1 : M$ **do**
$\tilde{\boldsymbol{X}}_m = \boldsymbol{J}\tilde{\boldsymbol{X}}_m$
6: **end for**
for $i = 1 : B$ **do**
8: • construct $\mathcal{T}_i'$: sample M scenarios with replacement from $\mathcal{T}$ in the high-level representation
• construct $\mathcal{T}_i$: in each scenario oversample the examples $(\tilde{\boldsymbol{x}}[n], \mathrm{y}[n])$ according to $\gamma[n]$
10: • construct the full-grown tree D_i by considering only $\sqrt{\tilde{N}}$ features at each split and by using the training set $\mathcal{T}_i$
end for
12: compute $R_{\text{oob}}(f(\boldsymbol{J}))$ where the loss is defined on scenario level and the mapping f is realized by the majority vote among the B trees
for $j = 1 : \tilde{N}$ **do**
14: remove the information from the jth feature and compute $R_{\text{oob}}^{(j)}(f(\boldsymbol{J}))$
compute $\Delta^{(j)} = R_{\text{oob}}^{(j)}(f(\boldsymbol{J})) - R_{\text{oob}}(f(\boldsymbol{J}))$
16: **end for**
choose a new $\boldsymbol{J}$ such that the feature with the minimal value of $\Delta^{(j)}$ is discarded
18: **end while**
the penultimate $\boldsymbol{J}$ is the desired selection matrix

$R_{\text{oob}}(f(\boldsymbol{J}))$ must be computed based on a loss function on scenario level as the one defined in Eq. (17). Similarly to the RF procedure for the usual feature-value model described in the previous subsection, also another loss function can be defined based on the margin of the SBRF algorithm, leading to an embedded method for feature selection in change detection. For this purpose we introduce the confidence based penalization

$$cp[n] = \gamma[n](1 - mg(\tilde{\boldsymbol{x}}[n], \mathrm{y}[n])), \tag{24}$$

where $mg(\tilde{\boldsymbol{x}}[n], \mathrm{y}[n])$ is defined analogously to Eq. (21) and is computed by using only those trees that have not used the scenario to which $\tilde{\boldsymbol{x}}[n]$ and $\mathrm{y}[n]$ belong during the training phase. The margin $mg(\tilde{\boldsymbol{x}}[n], \mathrm{y}[n])$ measures the degree of confidence one can have in the decision of the SBRF at time instance n. If for the input $\tilde{\boldsymbol{x}}[n]$ all trees in the SBRF identify the correct class $\mathrm{y}[n]$, then $cp[n]$ will be zero, whereas the greatest value of $cp[n]$ is $2\gamma[n]$ since the margin $mg(\tilde{\boldsymbol{x}}[n], \mathrm{y}[n]) \in [-1, 1]$. Whenever a misclassification occurs the value $cp[n]$ exceeds the value $\gamma[n]$.

Fig. 3 shows how $cp[n]$ evolves over time for a scenario from the application that will be described in the next section. Up to $n = 5$ the considered scenario corresponds to the class $c_1 = 0$, then a "do not care" interval follows until $n = 10$ and finally the corresponding class changes to $c_2 = 1$. As it can be seen from Fig. 3 the SBRF detects the class change only at $n = 13$, thereby leading to two misclassifications at $n = 11$ and $n = 12$. The confidence based penalization $cp[n]$ does not only penalize these two misclassifications but also the decisions at $n = 13, \ldots, 20$ since there are trees in the SBRF—as the margin $mg[n] = mg(\tilde{\boldsymbol{x}}[n], \mathrm{y}[n])$ shows—which decide for the wrong class.

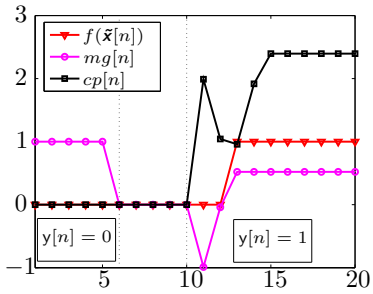

Fig. 3: Confidence Based Penalization $cp[n]$

Using the confidence based penalization we can define a loss function on scenario level that is typical for the SBRF-algorithm, namely

$$L(\boldsymbol{f}(\boldsymbol{X}), \boldsymbol{y}) = \frac{\sum_{n=1}^{n_{\text{end}}} cp[n]}{2\sum_{n=1}^{n_{\text{end}}} \gamma[n]}. \tag{25}$$

The loss $L(\boldsymbol{f}(\boldsymbol{X}), \boldsymbol{y})$ from Eq. (25) is the ratio of the areas below the confidence penalization $cp[n]$ and $\gamma[n]$ for the duration of a scenario. The lower $L(\boldsymbol{f}(\boldsymbol{X}), \boldsymbol{y})$ the more confidence we have that the decisions of the SBRF algorithm are correct for the scenario. Using this loss function in line 12 of Alg. 1 in order to estimate the risk $R(f(\boldsymbol{J}))$ we can perform feature selection based on the confidence information.

In addition to the reliable way to estimate the change detection performance by using the oob-method which is the basis for feature selection another advantage of the SBRF algorithm is the possibility—analogous to the RF algorithm [3]—to use categorical variables as features. This might play an important role when in the feature generation step one constructs features based on the similarity to a finite set of possible templates. Then a categorical variable might be used as a feature, indicating to which template the scenario is most similar.

V. Application

The methods that we introduced in the previous section can be applied to many industrial applications. In the following we will show how feature selection can be performed for car crash detection.

In modern cars there are various sensors used to detect and to categorize the severity of a crash. We will consider the task of identifying the relevant features that can be generated from

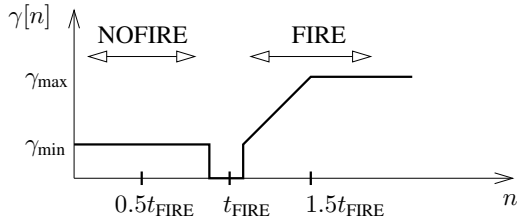

Fig. 4: $\gamma[n]$ for the Crash-Detection Application

the deceleration signals produced by 4 sensors during front-crash tests. Since the requirements set by law and consumer-tests like EURO-NCAP or US-NCAP are very demanding a simple threshold of a deceleration signal is not enough to determine the severity of a crash. Therefore, state-of-the-art algorithms for crash detection [27] use features which are deduced from the deceleration signals and define rules based on these features that fulfill the requirements for the available crash test measurements. Since the choice of the features and the rules are based on empirical experience we will apply the feature selection method to identify the most discriminative features.

Three of the four sensors measure the deceleration in longitudinal direction of movement at different locations in the car and one sensor the acceleration in the transversal direction. Since there are four time-series that have to be taken into consideration this is a typical change detection task in multivariate time-series where the data from each crash builds a scenario $\boldsymbol{S}$ containing $L = 4$ time-series. Depending on the crash severity it is required to activate the safety-systems, e. g., belt pretensioner or airbag at the right time. Thus, the output $y[n]$ belongs to one of the two classes $c_1 =$ FIRE or $c_2 =$ NOFIRE. For example the time instance where the class change occurs for the airbag deployment is computed using the "$5^{"} - 30$ ms" criterion. This criterion requires the deployment at the time instance where an unbelted passanger, being subject to the negative of the deceleration that is measured by one of the sensors—located in the passenger compartment—moves forward 5 inches minus 30 ms which are required for the inflation of the airbag. The class-change time t_{FIRE} is computed by subtracting 2 ms from the time computed using the "$5^{"} - 30$ ms" criterion. Then, on the left and the right of t_{FIRE} a "do not care" interval of 2 ms is introduced. This leads to a penalization $\gamma[n]$ as the one shown in Fig. 4. Not only a change detection that occurs too late but also a deployment that occurs too early must be penalized. The time-series taken into consideration have a duration of $t_{\text{FIRE}} + 20$ ms for all crashes that demand a deployment and the total measured length (ca. 100 ms) for misuse cases and harmless crashes that do not require the activation of any safety system. For these cases the penalization is set to $\gamma_{\min}$ for all time instances.

Using a data set of 83 independent scenarios $\boldsymbol{S}_m$, and a starting number of 53 features the above methods have been applied in order to determine good features for the car crash detection task. Among the 53 features that were generated from the deceleration signals there were traditionally used quantities like velocity loss or displacement but also features resulting from a time-scale (wavelet) decomposition of the signals. The SBRF considered here consist of $B = 300$ trees and the values $\gamma_{\max} = 5$ and $\gamma_{\min} = 1$ have been chosen.

No. features	estim. risk wrapper	estim. risk embedded	misclassified scenarios
53	0.24	$5.9 \cdot 10^{-2}$	2
33	0.29	$5.4 \cdot 10^{-2}$	2
25	0.30	$5.4 \cdot 10^{-2}$	2
15	0.37	$5.1 \cdot 10^{-2}$	1
10	0.36	$4.8 \cdot 10^{-2}$	1
8	0.32	$5.7 \cdot 10^{-2}$	1
5	0.40	$4.7 \cdot 10^{-2}$	1
4	0.33	$4.5 \cdot 10^{-2}$	1
3	0.39	$5.3 \cdot 10^{-2}$	2

TABLE I: Feature Selection with Wrapper Method

If the wrapper method is used, i. e., the feature selection is performed by using the loss from Eq. (17) we obtain as importances for the features the results from Table I. The first column in the table contains the number of features that were taken into account when constructing the corresponding SBRF. All results in a raw have been generated by the same SBRF. The second column shows the value of the oob-estimate of the risk when the loss from Eq. (17) is used and the third column shows the value of the oob-estimate computed with the loss from Eq. (25). The values in the third column have only been computed for the comparison with Table II. The fourth column presents the number of crashes that were misclassified over the whole duration of the crash, i. e., either a crash requiring the deployment of a safety system has been classified as a harmless crash, or a harmless crash led to the deployment of a safety-system. The critical value of the estimated risk from line 3 of Alg. 1 was set to 0.45. The crash that appears as misclassified scenario in the table when using the selected $4, 5, 8, 10$ or 15 features, represents a misuse case—the drive through a deep road hole—whose deceleration signals have been multiplied by the factor 1.3.

No. features	estim. risk wrapper	estim. risk embedded	misclassified scenarios
53	0.24	$5.9 \cdot 10^{-2}$	2
33	0.29	$5.6 \cdot 10^{-2}$	2
25	0.32	$5.8 \cdot 10^{-2}$	2
15	0.34	$6.0 \cdot 10^{-2}$	2
10	0.40	$5.2 \cdot 10^{-2}$	2
8	0.45	$6.0 \cdot 10^{-2}$	3
5	0.48	$6.1 \cdot 10^{-2}$	2
4	0.49	$6.4 \cdot 10^{-2}$	3
3	0.57	$6.2 \cdot 10^{-2}$	2

TABLE II: Feature Selection with Embedded Method

Applying the embedded method, i. e., the feature selection is performed by using the loss from Eq. (25) which is based on the confidence information indicated by the SBRF, we obtained Table II. Here, again 300 trees were used in the SBRF and the critical value of the estimated risk from line

3 of Alg. 1 was set to $6.5 \cdot 10^{-2}$.

Comparing Table I with Table II one sees that in this application the embedded method performs worse than the wrapper method in the feature selection task. Whereas the wrapper method leads to a reliable classification with only 4 features one would decide to use more than 10 features when using the embedded method. The explanation for this result is related to the reason why the loss in Eq. (17) has been preferred to the one from Eq. (19). Due to the correlation of the input vector $\boldsymbol{x}[n]$ to the inputs in his vicinity, adjacent misclassifications often have the same reason. Since we are trying to penalize this reason and not how long its effect lasts, the loss from Eq. (25) puts too much emphasis on long lasting intervals of uncertainty that have the same reason, i. e., intervals with strongly correlated inputs. For the example presented in Fig. 3 the region of the input space to which the inputs $\tilde{\boldsymbol{x}}[n]$ belong changes considerably between $n = 11$ and $n = 13$. Because the inputs following $n = 13$ are strongly correlated but have a very high $cp[n]$ value, a feature which influences the SBRF-decision in that region of the input space where these inputs lie, will wrongly get a too high weight compared to a feature which influences the region of the input space where $\tilde{\boldsymbol{x}}[n]$ lies at $n = 11$. Thus, a possible improvement for the loss function from Eq. (25) can be achieved by considering the change in the confidence penalization $cp[n]$ only in small intervals around the time instances where a class-change occurs either in $\boldsymbol{y}$ or in $\boldsymbol{f}(\tilde{\boldsymbol{X}})$.

After having identified the strongest features, interpretable classifiers like a single decision tree can be trained, such that the resulting rules can be validated by experts. Thereby, in order to take account for the fact that we are dealing with time-series one must use a risk defined on scenario level.

VI. Conclusion

Two methods how feature selection can be performed in change detection applications have been presented in the paper. The first can be used with every machine learning algorithm, whereas the second method makes use of the confidence one has into a decision and is specific for the SBRF algorithm. The key for feature selection in time-series is a reliable estimation of the generalization error which has to be done on scenario level and not on time-stamp level. Identifying the relevant features for an application offers the possibility to use these features for training a decision tree, thereby enabling interpretability. Thus, numerous applications can benefit from the feature selection methods described in the paper. As an example a car crash detection application has been presented. Further analysis of the method is the subject of current research as well as the use in other change detection applications.

Acknowledgment

This work is partially a result of the collaboration on machine learning applications in car safety systems with the AUDI AG in Ingolstadt, Germany. We thank the company for providing the car crash data set. We also thank Dr. Frank Keck and Dipl.-Ing. Christian Weiß for the fruitful discussions leading to this work.

References

[1] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, Prentice Hall, 1994.

[2] R. Bellman, *Adaptive Control Processes: A Guided Tour*, New Jersey: Princeton University Press, 1961.

[3] L. Breiman, "Random forests", *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[4] M. W. Kadous, "Learning comprehensible descriptions of multivariate time series", in *Proceedings of the 16th International Conference on Machine Learning, ICML'99*, Bled, 1999, pp. 454–463.

[5] C. J. A. González and J. J. R. Diez, "Time series classification by boosting interval based literals", in *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 2000, pp. 2–11.

[6] P. Geurts, *Contributions to decision tree induction: bias/variance tradeoff and time series classification*, PhD thesis, University of Liège, Belgium, 2002.

[7] M. W. Kadous and C. Sammut, "Classification of multivariate time series and structured data using constructive induction", in *Machine Learning*, 2005, vol. 58, pp. 179–216.

[8] P. Geurts and L. Wehenkel, "Segment and combine: a generic approach for supervised learning of invariant classifiers from topologically structured data", in *Machine Learning Conference of Belgium and The Netherlands*, 2006, pp. 15–23.

[9] P. Geurts, "Pattern extraction for time series classification", in *Principles of Data Mining and Knowledge Discovery, 5th European Conference, PKDD*, Freiburg, Germany, 2001, pp. 115–127.

[10] C. S. Myers and L. R. Rabiner, "A comparative study of several dynamic time-warping algorithms for connected word recognition", in *The Bell System Technical Journal*, 1983, vol. 607, pp. 1389–1409.

[11] E. Keogh and M. Pazzani, "Dynamic time warping with higher order features", in *SIAM International Conference on Data Mining*, Chicago, USA, 2001.

[12] I. Mierswa and K. Morik, "Automatic feature extraction for classifying audio data", in *Machine Learning*, 2005, vol. 58, pp. 127–149.

[13] R. T. Olszewski, *Generalized feature extraction for structural pattern recognition in time-series*, PhD thesis, Carnegie Mellon University, Pittsburgh, 2001.

[14] B. Rice, "Techniques for developing an automatic signal classifier", in *Proceedings ICSPAT*, 1999.

[15] C. A. Ratanamahatana and E. Keogh, "Making time-series classification more accurate using learned constraints", in *Proceedings of SIAM*, 2004.

[16] D. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series", in *AAAI Workshop on Knowledge Discovery in Databases*, 1994, pp. 229–248.

[17] L. Breiman, "Out-of-bag estimation", Tech. Rep., University of California, Berkeley, 1996.

[18] R. J. Tibshirani, "Bias, variance and prediction error for classification rules", Tech. Rep., University of Toronto, November 1997.

[19] D. H. Wolpert and W. G. Macready, "An efficient method to estimate bagging's generalization error", Tech. Rep., Santa Fe Institute, 1996.

[20] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, The Wadsworth & Brooks/Cole Statistics/Probability Series. Wadsworth, 1984.

[21] P. Geurts, D. Ernst, and L. Wehenkel, "Extremly randomized trees", *Machine Learning*, 2006.

[22] J. H. Friedman, "On bias, variance, 0/1-loss and the curse of dimensionality", *Data Mining and Knowledge Discovery*, vol. 1, pp. 55–77, 1997.

[23] L. Breiman, "Manual on setting up, using, and understanding random forests v3.1", 2002.

[24] C. Chen, A. Liaw, and L. Breiman, "Using random forest to learn imbalanced data", Tech. Rep., University of California, Berkeley, 2004.

[25] P. Domingos, "Metacost: A general method for making classifiers cost-sensitive", in *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, 1999, pp. 155–164.

[26] C. Ling and C. Li, "Data mining for direct marketing problems and solutions", in *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, New York, 1998.

[27] M. Huang, *Vehicle Crash Mechanics*, CRC Press, 2002.